Postel

Network Working Group
Request for Comments: 44

A. Shoshani
R. Long
A. Landsberg
System Development Corporation
10 April 1970

Comments on NWG/RFC 33 and 36

Generally, we are satisfied with the suggestions for the new Host-to-Host protocol. However, we think that a few refinements may be helpful.

I. It seems that there are two cases of reconnection:

1. Reconnect from a socket in a local Host to another socket in the local Host. This was referred to in RFC #33 as "switch". The local sockets can belong to different processes (such as the "Login" process switching a connection to another process just created) or can belong to the same process (such as a process that accepts calls for connections on a particular socket, and after a connection is established switches to another of his sockets).

2. Reconnect from a socket at a local Host to a socket in a foreign Host.

We suggest separation of these two cases for the following reasons:

a) Reconnection in Case 1 is necessary and useful, while the usefulness of Case 2 is still in doubt.

b) Case 1 is simple to implement (at least conceptually) while Case 2 involves an elaborate mechanism of commands because of the asynchronous nature of the network (four out of nine commands were suggested to handle Case 2 in RFC #36).

Thus we think that at least in the first usage of the Host-to-Host protocol reconnection in Case 2 should be left out. An additional system call (not a command) is therefore needed to permit Case 1, which is: SWITCH <socket 1> <socket 2>.

II. The CLOSE command as suggested in RFC #36 seems to be used for two purposes: block a connection and abort a connection. To avoid ambiguity it would be desirable to have two commands: BLOCK and

CLOSE. As suggested in RFC #36, the response for both commands can be the SUSPEND command which acknowledges the reception of BLOCK or CLOSE commands.

III. After a connection has been established, we see no reason for keeping the "foreign socket" in a local connection table. Since there is a one-to-one correspondence between a link number of the foreign Host and a foreign socket number, we can use the link number in the commands. Thus, except for the RFC command, all commands can use link numbers and therefore eliminate a 40-bit foreign socket number in every entry of the connection table (size being critical for some Hosts). We note that if connections will be multiplexed over links as suggested in RFC #38, then the foreign socket would be needed in the connection table.

IV. In RFC #33 the term PORT was introduced. Although this is private to every Host, we have a comment. If ports are used such that there is a one-to-one correspondence between a port for some user and a socket, then ports are completely redundant. However, a Host may wish to multiplex ports over connections, in which case an additional mechanism is needed.

To summarize the last four comments, we suggest that in the initial version the following system calls and commands will be used (most of them as in RFC 33 and 36).

System Calls:

1) INITIATE <my socket> <your socket>
2) ACCEPT <my socket>
3) SWITCH <socket 1> <socket 2>
4) LISTEN <my socket>
5) CLOSE <my socket>
6) TRANSMIT <my socket> <address>

Commands:

Commands 0, 1, 3, 4 as in RFC #36 (pp.5) and in addition:

1) BLOCK: BLK <link>
2) CLOSE: CLS <link>

V. In addition to the above it seems necessary to decide on the following issues one way or the other together with the first version of the protocol (perhaps by setting a date for people to express their preferences and decide accordingly). All of these issues were mentioned in the meeting at UCLA on March 17, 1970, but were put aside.

1. "Double padding" - when a message does not end on a word boundary. Two possible solutions were mentioned:
   a) Hosts provide their padding in addition to the IMP's padding (double padding).
   b) Hosts make sure that all messages end on a word boundary by shifting their messages (when necessary) and adjusting the "marking" accordingly.
2. "Echoing" - there are three apparent possibilities:
   a) Echoing
   b) No echoing
   c) Optional Echoing - possibly a bit in the "Leader" can be used to designate this option.
3. "Code Conversion" - originally, BB&N suggested doing the conversion in the IMPs using ASCII-8 as the common code. This was rejected, mainly because of claims that ASCII-8 is not large enough for some uses, such as graphics. Also, conversion in the IMPs may slow them down and take up space which could be used for buffers. We feel that it is very desirable to have a common code (even when the conversion is not done by the IMPs), such that all incoming text messages are in the same code and only one conversion table is needed. Outgoing text messages should be converted into this common code. Obviously, the option "no translation" should be possible for the purpose of binary data or data that is not representable in the common code.

Since every known code can be considered to be too restrictive for some purposes, we suggest adopting a Network Common Code (NCC), and use all of the 256 possible characters (for 8-bit code) to include the "important" part of the union of the codes used throughout the network.

VI. Our preference to the above issues is as follows:

a) "Double padding" - it turns out to be easy for us to get our messages to be sent on a word boundary by shifting the leader of a message (and adjusting the "marking" accordingly) rather than the data. Thus we will prefer solution V.1.b).

b) "Echoing" - we prefer no echoing. We think that character echoing should be managed locally.

c) "Code Conversion" - we prefer a Network Common Code. Initially, ASCII-8 can be used, and then expanded according to the needs of the Network.